Silversite Developers Overview

July 26, 2012

1 Introduction

1.1 Motivation

Why Silversite? Why yet another ASP.NET CMS? The motivation for the development of Silversite was, that existing ASP.NET CMS had too slow start up times, and did not integrate well with ASP.NET. Take for example DotNetNuke. I implemented a site for our society with DotNetNuke. The time for the first page to load on application start up was around 13 seconds. How do you explain this to customers? Especially if you love ASP.NET and want to convince people to use an ASP.NET CMS and not a PHP one? Or take Sitefinity. A beautiful design, but slooooow. A ton of dll's in the bin folder. This way you will never get reasonable start up times. Another point of frustration was that if you are used to ASP.NET development, then when you use a CMS, you cannot use the full power of ASP.NET anymore and must stick to plain HTML and the offered modules. The motivation to create Silversite grew out of this frustration and out of the ideas on how to make it better. Therefore Silversite tries to make it better.

1.2 Design Goals & Concepts

So the main design goals with Silversite were speed and ASP.NET integration. Now, how do you improve the speed of a CMS? First there are two places, where speed matters. One is application start up time. The other is page response time.

Application start up time, in ASP.NET, is correlated to the number and size of dll's contained in the bin folder of the application, because that dll's need to be loaded and compiled every time your application starts. But the normal viewer of your web site might actually only need a fraction of the code that's loaded at start up. For example, an ordinary site visitor will never use the code that is related to editing the CMS content. This code need only to be loaded if an administrator log's in. So one of the concepts of Silversite is, that only those dll's that are really needed are placed in the application's bin folder. The optional dll's reside in another folder and are loaded as needed.

Now, how do you improve page response time? An ordinary CMS stores the dynamic content of it's modules in a database. This way, each time the page is accessed, the content must be retrieved from the database. But why not store the dynamic content directly in the page itself? Of course it is not so trivial to store the content directly in the page. You need an ASP.NET Parser that parses your page into a DOM structure, so you can insert the content in the correct place. That's exactly what Silversite does. So Silversite does not access any database for most page requests. There is another very important advantage of storing the content directly in the page. It now becomes possible to have not just plain HTML but also ASP.NET controls in your content, because the content is in the page and get's compiled by the ASP.NET page compiler. So this allows for ASP.NET integration in the editable content of the CMS.

2 Implemented Features

2.1 Service / Provider Model

Silversite is build to be extensible. Therefore every extensible concept in Silversite is implemented as a Service. Every Service can have multiple Providers that provide an implementation of the Service. For example there is a JavaScriptHtmlEditor Service, and a provider for the CKEditor html editor.

2.2 Html Processing

Silversite needs to be able to process ASP.NET code, in order to save content in aspx pages. Therefore Silversite implements classes for parsing, writing and converting such code. For example you might want to implement special editing features for editing ASP.NET controls, e.g. like in the VisualStudio designer. So the html you display in the editor might be different than the html you save as content and needs to be converted back and forth for editing. There is a special class Silversite.Services.Html.Converter for this purpose, and the Provider for the CKEditor provides such a converter.

Or another example might be, that you extend the basic EditableContent control that allows for dynamic content in a page, because you don't want to edit html but maybe wiki code. Also here a converter can convert from the saved content, where the wiki code might be stored in a server side comment, to the wiki code that get's edited in the editor, and convert back again to ASP.NET after the wiki code has been edited.

2.3 Database Access

Silversite uses Entity Framework CodeFirst for database access. Unfortunately, the current implementation of CodeFirst does not support extensibility, because it allows only one DbContext per database, and Silversite needs to be able to have multiple DbContext on the same database for extensions. So the current Silversite temporary implementation only has one DbContext and is not extensible.

2.4 Support for multiple Domains

Silversite supports multiple domains. It has a special folder, where the pages for the domains get stored. This folder contains a sub-folder for each supported domain and a folder “default” for the default pages, that are visible in all domains.

2.5 VirtualPathProvider for Resource files

Silversite extensions will usually reside in a custom dll. Normally extensions also need pages, and the pages would be deployed in the file system. Silversite implements a VirtualPathProvider, so that all the files in your dll you marked as Resource and that reside in special folders like “silversite” or “App_Virtual” can be accessed, as if they would reside in the file system.

2.6 Backups

Silversite provides for easy site backups. This feature is not yet implemented, only the core concept. The concept is, that every Service that has data associated to it implements the IDataService, that has methods to read and write the Service's data from and to a Stream. This way, all data in Silversite can be stored in a backup file.

2.7 File System Access

In order to make it easy to configure Silversite for write access in a shared hosting environment, all stuff in Silversite is contained in the silversite folder, that needs write access on the server. Other stuff might be contained in an ASP.NET folder like bin or in the root directory like the web.config file. In order to support shared hosting best, Silversite can use FTP for file system write access, where it doesn't have write permission otherwise.

2.8 EditableContent control

This is the base class for all editable content in a control. It implements storing the content in the aspx page, and common ui stuff for editable content. It is also extensible to support different content and different content editors.

3 Planned Features

3.1 CKEditor adaptions

The CKEditor still needs a plugin for a toolbar item that allows the insertion of ASP.NET stuff and CMS Modules.

3.2 File Manager

A JavaScript File Manager for the CKEditor and for the CMS is under development. Maybe it would be worth it, to implement the File Manager in Silverlight, to allow access of the client file system and drag & drop.

3.3 Administration Pages

No development has yet been done on any pages for CMS administration. Anyway those pages will be implemented using the CMS, and users will be able to edit those pages if needed using the CMS. Maybe it would be an option to implement the Administration in Silverlight.

3.4 Modules

We haven't developed any modules yet. The goal is, that modules are treated as ordinary ASP.NET Server Controls, and derive from the EditableContent control to implement basic functionality.